TOPOLOGY PROCEEDINGS
Volume 12, 1987
Pages 117–158
http://topology.auburn.edu/tp/

TOPOLOGY OF COMPUTER VISION

by

GERHARD X. RITTER

TOPOLOGY PROCEEDINGS Volume 12 1987

TOPOLOGY OF COMPUTER VISION

Gerhard X. Ritter

1. Introduction

The principal objective of this paper is to provide an introduction to some basic concepts and techniques in the domain of computer vision, and to focus attention on several diverse applications of topology to this novel discipline. These applications overlap the author's own areas of interest and research. Thus, this paper should not be viewed as an all encompassing survey of the applications of topology to the field of computer vision. However, we do hope that this paper will attract the attention of topologists interested in applying their knowledge to the many problems that exist in the field of computer vision. The topics and references listed in the last section of this paper should provide a good introduction to these problems.

The field of computer vision and its major subdisciplines--image processing, pictorial pattern recognition and image understanding--has grown considerably during the past decade due to the increased utilization of imagery in medical, industrial, space and military applications. The principal application areas are the improvement of pictorial information for human interpretation and the processing of pictorial data for autonomous machine perception. Improvement of pictorial information for human interpretation include the resolution improvement and noise filtering in X-ray tomography imagery, the compensation of sensor and transmission errors of pictures transmitted from deep-space probes, and the improvement of edge information in low resolution infrared images.

Typical routine applications in machine perception are automatic character recognition, automatic processing of fingerprints, automatic morphological classification of blood cells, and the automatic processing of satellite imagery for weather prediction and militpry recognizance.

2. Image Representation

Computer images are modeled in terms of continuous physical images. To be more explicit, let $\ell(x,y,t,\lambda)$ represent the spatial energy distribution of an image source of radiant energy at spatial coordinates (x,y), time t and wavelength λ. Since the light function is non-negative and real, and the physical imaging system imposes some restriction on the brightness of the image, it is assumed that $0 \leq \ell(x,y,t,\lambda) \leq c$ for some constant c. Furthermore, as a scene is observable only over some finite time interval, the light function is a bounded function with three bounded independent variables.

In an imaging system, the observed image field is modeled as a spectrally weighted integral of the image light function:

$$a(x,y,t) = \int_0^\infty \ell(x,y,t,\lambda)s(\lambda)d\lambda$$

where $s(\lambda)$ denotes the spectral response of the sensor.

In many--but not all--imaging systems, the time variable is dropped as the image (e.g. photograph) does not change with time.

A *digital image* a(i,j,k) is viewed as the discretized version of a sampled, continuous image field a(x,y,t) in space, time and intensity. When manipulating digital images by computer for such purposes as edge detection or analysis in the Fourier domain, the positive integral values a(i,j,k) may be transformed into negative, real or complex numbers. In multisensor data analysis, data fusion algorithms often view an image a(i,j,k) as

$$a(i,j,k) = (a_1(i,j,k),a_2(i,j,k),\cdots,a_n(i,j,k)),$$

where

$$a_i(x,y,t) = \int_0^\infty \ell(x,y,t,\lambda)s_i(\lambda)d\lambda$$

and $s_i(\lambda)$ denotes the spectral response of the i-th sensor. In addition, spatial coordinates may not be restricted to 2-dimensional planar coordinates. In lasar-radar imaging, spatial coordinates are 3-dimensional since the returning signal provides for range information. It is these observations that provide the model and basis for a general standard definition of a *computer* image.

Henceforth let Z, R, C, and Z_{2^k} denote the sets of integers, real numbers, complex numbers, and binary numbers of fixed length k, respectively. For $n \in Z$ and $n \geq 0$, let R^n denote n-dimensional Euclidean space.

Definition 2.1. Given a compact set $X \subset R^n$ and a groupoid F with identity, then an F *valued image* A *on* X is the graph of a function $a: X \to F$, i.e. $A = \{(x,a(x)): x \in X\}$. The set of all F valued images on X is denoted by F^X. An element $(x,a(x)) \in A$ is also called a *pixel* (=picture element) of A and a(x) the *gray value* or *gray level* at location x.

If the groupoid $F = R$ or $F = R^n$, then we are dealing with real valued or n-dimensional vector valued images, respectively. Similarly, replacing F by Z, C, or Z_{2^k}, provides for integral, complex or finite digital images, respectively. These are the most commonly used value sets in image processing. It is also often convenient to replace R by $\bar{R} = R \cup \{-\infty,\infty\}$ and allow extended arithmetic and logic operations. This extension is especially useful when manipulating "raw" radar images which, due to sensor recording errors, have spatial locations with "out-of-range" values and locations with no signal values, called *missing values*. These locations can be assigned special symbols corresponding to ∞ and $-\infty$.

In the remainder of this exposition, the term "image" shall always mean an image as defined above. Unless otherwise stated, we shall also assume that $F = R$, and X is a finite subset of R^n. This will facilitate and retain the discussion on the intended introductory level.

By defining the "right" algebraic operations, images can be manipulated and behave very much like real numbers. In fact, the basic image operations reflect the arithmetic and logic operations on R. In particular, the binary operations of addition, multiplication, maximum, and exponentiation on R^X are defined as follows:

Let $A, B \in R^X$. Then

1. $A + B \equiv \{(x,c(x)): c(x) = a(x) + b(x), x \in X\}$
2. $A * B \equiv \{(x,c(x)): c(x) = a(x) * b(x), x \in X\}$
3. $A \vee B \equiv \{(x,c(x)): c(x) = a(x) \vee b(x), x \in X\}$

4. $A^B \equiv \{(x,c(x))\colon c(x) = a(x)^{b(x)}$ if $a(x) \neq 0$, else $c(x) = 0, x \in X\}$.

   We restrict this binary operation to those pairs of images A,B for which $a(x)^{b(x)} \in R$. The inverse of exponentiation is defined in the usual way by taking the logarithm. In particular, we define:

5. $\log_A B \equiv \{(x,c(x))\colon c(x) = \log_{a(x)} b(x), x \in X\}$.

   As for real numbers, $\log_A B$ is defined only for those images A and B for which $a(x) > 0$ and $b(x) > 0$ for all $x \in X$.

The next basic binary operation, called the *dot product*, distinguishes itself from the above five in that its output is not an image but a real number.

6. $A \cdot B \equiv \sum_{x \in X} a(x)b(x)$

Definition 2.2. An image A is called a *constant image* if all its gray values are the same; i.e. if $a(x) = k$ for some real number k and for all $x \in X$.

Two important constant images are the *zero* image, defined by $0 \equiv \{(x,0)\colon x \in X\}$, and the *unit* image, defined by $I \equiv \{(x,1)\colon x \in X\}$.

Suppose $k \in R$ and A is a constant image with $a(x) = k$. Then we define:

i. $B^k \equiv B^A$ and $k^B \equiv A^B$

ii. $kB \equiv A * B$ and $\dot{k} + B \equiv A + B$

iii. $\log_k B \equiv \log_A B$, of course $k > 0$ and $b(x) > 0$ for all x.

We note that exponentiation is defined even when $a(x) = 0$. Subtraction, division and minimum are defined in terms of the basic operations and inverses. Specifically:

iv. $A - B \equiv A + (-B)$ and $A/B \quad A * B^{-1}$, where

$-B = \{(x,-b(x)) : (x,b(x)) \in B\}$

v. $A \wedge B \equiv -(-A \vee -B)$

The images 0 and I have the obvious property $A + 0 = A$ and $A * I = A$. On the other hand, $B * B^{-1}$ does not necessarily equal I. However, $B * B^{-1} * B = B$. For this reason B^{-1} is called the *pseudo* inverse of B. Inequalities between images are defined in terms of maximum and minimum. Thus, for example, $A \leq B$ if $A \vee B = B$. These observations show that the ring $(R^X,+,*)$ and the lattice $(R^X,\vee,\leq)$ behave very much like the ring and lattice of real numbers.

The *complement* of an image A is denoted by $\bar{A}$ and is defined as $\bar{A} = I - A * A^{-1}$. The definition of *characteristic value* makes use of the concept of complementation. In particular, if A and B are images, then

$$c_{>B}(A) = [(A - B) \vee 0]^{-1} * [(A - B) \vee 0]$$

Thus,

$$c_{>B}(A) = \{(x,c(x)) : c(x) = 1 \text{ if } a(x) > b(x), \text{ else } c(x) = 0\}.$$

The remaining characteristic functions of images can be defined in a similar fashion, using complementation and products. For example,

$$c_{\leq B}(A) = \overline{c_{>B}(A)}$$

and

$$c_B(A) = c_{\leq B}(A) * c_{\geq B}(A)$$

Whenever B is the constant image with gray values equal to k it is customary to replace B by k in the above definitions. Note that the characteristic function provides a good example as to the use of the pointwise maximum and minimum operations.

3. Cellular Topology

The cellular automata of von Neumann and Moore and computer image manipulation share a common framework [1,2]. Each point $(x,a(x)) \in A$ can be viewed as a point $x \in X$ in a given state $a(x)$. Be defining neighborhood relationships and cell transition functions--also known as *template functions* and *template operators*--the state of a cell can be changed to a new state, where the new state depends on the states of the cells in its neighborhood. Because of the dependence of these transition functions on a cells' neighborhood, these functions are also known as *neighborhood transforms*.

We begin by defining the concept of a generalized template function. Recall that a function $\mathcal{f}$ from a set Y to a cartesian product $X \times Z$, $\mathcal{f}: Y \to X \times Z$, induces a pair of *coordinate functions* $\mathcal{f}_1, \mathcal{f}_2$ called the coordinates of $\mathcal{f}$. More precisely, $\mathcal{f} = (\mathcal{f}_1, \mathcal{f}_2)$, where for each $y \in Y$, $\mathcal{f}(y) = (\mathcal{f}_1(y), \mathcal{f}_2(y))$, with $\mathcal{f}_1(y) \in X$ and $\mathcal{f}_2(y) \in Z$.

Definition 3.1. Let X,Y be compact subsets of R^k and R^n, respectively, and $A \in F^X$. A *generalized* F-*valued template* from Y to X is a function $T = (\mathcal{J},t): Y \to 2^X \times F^X$ whose second coordinate satisfies the property

$$t_y = \{(x, t_y(x)) : t_y(x) = 0 \text{ if } x \notin \mathcal{J}(y)\},$$

where $t_y \equiv t(y)$, and 0 denotes the identity of F; i.e. the support of t_y lies in $\mathcal{J}(y)$. The point y is called the *target point* of the *source configuration* $\mathcal{J}(y)$, and the values $t_y(x)$ for $x \in \mathcal{J}(y)$ are called the *weights* of T(y).

Given a template $T = (\mathcal{J}, t)$, then T is called a *template function with configuration* $\mathcal{J}$, and $\mathcal{J}$ is called a *source* or *neighborhood configuration of* Y *on* X. If Y = X, then $(\mathcal{J}, t)$ is simply called a *template on* X and $\mathcal{J}$ a *neighborhood configuration on* X.

For real valued templates and images, there are three basic template or neighborhood operations which are used to transform an image. They are denoted $\oplus$, $\circledvee$, and $\boxed{\vee}$. These neighborhood operations transform each image point by performing the basic operation of addition or maximum on a weighted collection of neighboring image values. In particular, if $A \in R^X$ and T is a template from Y to X, with $\mathcal{J}(y)$ finite for each $y \in Y$, then

$$A \oplus T \equiv \{(y, c(y)) : c(y) = \textstyle\sum_{x \in \mathcal{J}(y)} a(x) t_y(x), \text{ where } y \in Y\}$$

$$A \circledvee T \equiv \{(y, c(y)) : c(y) = \vee_{x \in \mathcal{J}(y)} a(x) t_y(x), \text{ where } y \in Y\}$$

$$A \boxed{\vee} T \equiv \{(y, c(y)) : c(y) = \vee_{x \in \mathcal{J}(y)} a(x) + t_y(x), \text{ where } y \in Y\}$$

The complementary minimum operations are defined by $A \circledwedge T = -(A \circledvee -T)$ and $A \boxed{\wedge} T = -(-A \boxed{\vee} -T)$.

The operands (images and templates) and operators $(+, *, \vee, \oplus, \circledvee, \boxed{\vee})$ listed above define a heterogeneous algebra which provides a uniform mathematical environment to manipulate and transform computer images in order to compress or smoothen data, identify, classify and/or track objects, or to perform other desired tasks. A typical sequence of these manipulations, called an *image processing algorithm*, may consist of noise filtering, thresholding and background removal. This sequence may then be followed by such processes as thinning, edge detection or skeletonizing in order to obtain shape descriptors and/or achieve data compression. The geometric properties inherent in the transformed objects usually serve as a basis for object classification. Of special importance are such topological properties as nearness, connectivity, path-connectivity, genus, homotopy, and dimension.

Whenever such notions as connectivity, genus and homotopy are considered, topologies must be defined on the set of spatial coordinates. For the ensuing discussion, let $X \subset Z^k$, where Z^k denotes the k-fold cartesian product of Z. Any topology on X is called a *cellular* topology [3]. A commonly used topology is the *von Neumann topology* defined as follows:

Let $J = \{-1,0,1\}$ and $x \in X$. The neighborhood N(x) is defined by

$$N(x) = \begin{cases} \{x\}, \text{ if } \sum_{i=1}^{k} x_i \text{ is odd} \\ \{(x_1,\cdots,x_{i-1},x_{i+j},x_{i+1},\cdots,x_k): \\ \qquad 1 \leq i \leq k,\ j \in J\}, \text{ otherwise} \end{cases}$$

The collection $\mathcal{N} = \{N(x): x \in X\}$ is a neighborhood basis for the von Neumann topology on X.

In order to simplify our discussion, we first consider the case when $k = 2$. In this case, a connected set in the von Neumann topology is also called a 4-*connected* set since any point (cell) $\mathbf{x} = (x,y)$ is connected to each of its four horizontal and vertical neighbors; namely $(x \pm 1,y)$ and $(x,y \pm 1)$, respectively. This topology was first described by A. Rosenfeld [4]. Observe that the set $\{(x,y),(x + 1,y + 1)\}$ is not connected. There are various topologies such that for each point (x,y) and for every pair $i,j \in J$, the pair $(x,y),(x + i,y + j)$ forms a connected set. However, there does not seem to exist an example of a "finest" topology $\mathcal{J}$ for X (i.e., one having the smallest $\max\{|N(x)|: N(x) \in \mathcal{N}$ where $\mathcal{N}$ is a neighborhood basis for $\mathcal{J}\}$) which provides for 8-connectivity.

Although the von Neumann topology satisfies only the weak T_0 separation axiom, it "models" many common and important geometric and topological properties of Euclidean 2-space surprisingly well. The subsequent examples are a case in point. Proofs and a more detailed treatment of these examples and other related material can be found in the referenced literature.

A 4-*neighbor* or, simply, a *neighbor* of a point $\mathbf{x}$ (in the von Neumann topology) refers to one of its immediate vertical or horizontal neighbors. The set of 8-*neighbors* of $\mathbf{x} = (x,y)$ consist of its 4-neighbors together with the diagonal neighbors $(x + 1,y \pm 1)$ and $(x - 1,y \pm 1)$.

Let $B \subset X$ and, in order to avoid special cases, assume that B does not intersect the boundary of X. Then B is called an *arc* if B is connected, contains two points which have exactly one neighbor in B and any other points in B have exactly two neighbors in B.

Let E(x) denote the set of 8-neighbors of x but not including x. A point $x \in B$ is called a *simple point* of B if $B \cap E(x)$ has the same number of components as $B \cap (E(x) \cup \{x\})$. The following theorem was proven by A. Rosenfeld [4].

Theorem 3.1. B *is an arc if and if it is simply connected and has exactly two simple points.*

Arcs, simple closed curves and bouquets of simple closed curves in image processing are obtained from thinning wide objects into idealized thin forms which are then used in shape analysis or data reduction schemes. A connected set B is called a *simple closed curve* if

(i) each of its points has exactly two neighbors in B and

(ii) if a point x of B has a diagonally adjacent neighbor (an 8-neighbor which is not a 4-neighbor) y in B, then one of the two neighbors common to both x and y must also be in B.

If only condition (i) holds, then B is commonly known as a *curve*.

Theorem 3.2. B *is a simple closed curve if and only if* B *is connected, separates* X *into two components and* B *has no simple points.*

The proof of this theorem can be derived by appropriately modifying Rosenfeld's curve classification theorem and taking condition (ii) into account. We also note that connected is equivalent to path-connected and that digital simple closed curves in Z^2 have the same behavior in terms of separation properties, homotopy and genus as do simple closed curves in the plane. Similar observations hold for digital arcs and lines [4,5].

Although the von Neumann topology is a very elementary topology, defining efficient algorithms for the extraction of simple topological features or geometric measurements is far from trivial. Consider the case of defining an algorithm that computes the Euler characteristic of an object in an image. In addition, suppose that the algorithm design should be such that its implementation computes the Euler characteristic in terms of local knowledge of the object under consideration--i.e. new pixel values can only be computed in terms of neighboring pixel values. Before presenting an example of an algorithm satisfying these conditions, we briefly outline a computer architecture which: (i) is capable of implementing the algebraic operations described earlier, (ii) models the von Neumann and 8-connected topologies; and (iii) provides the rationale as to why one would like to consider "local" image processing schemes.

About 25 years ago, Unger proposed that many algorithms for image processing and analysis could be implemented in parallel using "cellular array" computers [6]. These cellular array computers were first inspired by von Neumann [7]. Von Neumann envisioned arrays of thousands of processing elements connected in such a fashion that each processing element could communicate with its directly adjacent neighboring processors in a square (or hexagonal) tessellation. Recent advances in VLSI technology finally permitted the realization of such arrays. Cellular computers for image processing are now in use in hundreds of laboratories worldwide. NASA's massively parallel processor or MPP [8], Martin Marietta's GAPP II+ [9], and the CLIP series of computers developed by Duff [10], represent the classic embodiment of von Neumann's original cellular automation. The CLIP4 consists of an array of 9216 (96×96) processors with sets of eight processors integrated on a single chip. The MPP also integrates eight processors per chip in an assemblage of 128×132 processing elements. In distinction to the CLIP, where each processing element has the capability of communicating with its eight immediate neighbors, an MPP processing element has connections to only four immediate neighbors as indicated by the solid lines in Figure 1.

Using these types of hardwired communication links between neighboring processors, each processor is responsible for one pixel and is capable of performing local operations on the image via its communication links. These local operations correspond to the previously defined algebraic operations and are performed in parallel on the whole image. Consequently, the operands for these operations are whole images and local templates, and each

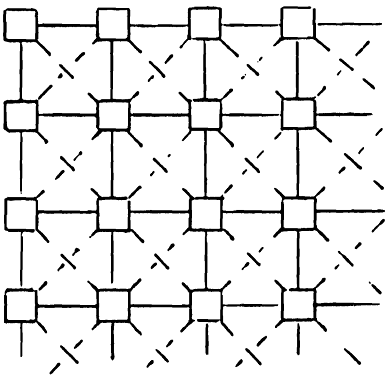


Figure 1. Cellular image processing automaton of identical processors with nearest neighbor connection.

operation is applied to a whole image as a *one-step real-time* operation. Hence these architectures provide for real-time or quasi real-time image manipulation [11]. In comparison, convolution-like operations on large arrays such as $\oplus$ on sequential machines are computationally intensive and extremely time consuming.

Given the availability of cellular array computers, we are faced with the new problem of writing algorithms that will take advantage of their architectures. The design of such algorithms is not necessarily an easy or straight forward task. This should become apparent in the computation of the Eular characteristic.

A *black and white* or *Boolean* image is an image in which a pixel has value 0 or 1. The set of pixels having value 1 is called the *black part* of an image. Now suppose A is a black and white image, B is the black part of A and the goal is to find the Euler characteristic of B. Consider the following set of 2×2 pixel patterns, called *bit quads*:

$$Q_1 = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 0 \\ \hline\end{array}\,, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 0 \\ \hline\end{array}\,, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline\end{array}\,, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 1 \\ \hline\end{array}$$

$$Q_2 = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline\end{array}\,, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 0 \\ \hline\end{array}$$

$$Q_3 = \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 0 & 1 \\ \hline\end{array}\,, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline\end{array}\,, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 1 & 1 \\ \hline\end{array}\,, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 0 \\ \hline\end{array}$$

The Euler characteristic χ of B can be expressed in terms of the number of bit quad counts of the image A by the formula

$$\chi = \frac{1}{4}[n(Q_1) + 2n(Q_2) - n(Q_3)]$$

The proof that this formula represents the Euler characteristic is non-trivial and employs R. Bott's critical point theory [12].

Since each bit quad pattern constitutes a "local" pattern, the formulation of χ in terms of bit quad counts simplifies the task of formulating an algorithm that satisfies the previously mentioned requirements. To begin with, let $x = (x,y)$ be an arbitrary point of X, $x_1 = (x,y - 1)$, $x_2 = (x + 1,y)$, and $x_3 = (x + 1,y - 1)$. Let T be the template defined by the weights $t_x(x) = t_x(x_3) = 3$, $t_x(x_1) = t_x(x_2) = 1$, and $t_x(y) = 0$ if $y \neq x, x_1, x_2$, or x_3. Thus, T(x) has configuration as shown:

$$T(x) = \begin{array}{|c|c|}\hline 3 & 1 \\ \hline 1 & 3 \\ \hline \end{array}$$

where the shaded cell represents the target point x of T(x).

The algorithm for computing χ can now be expressed by the one line algebraic formula

$$\chi = \frac{1}{4}[c_1(A \oplus T) + 2c_2(A \oplus T) + c_3(A \oplus T) - c_5(A \oplus T) + 2c_6(A \oplus T) - c_7(A \oplus T)] \cdot I$$

This formula will be accepted by an appropriate translator as a valid computer code. It will be instructive to examine the algebraic formulation representing χ. We first note that overlaying T with one of the Q_1 bit quad patterns and computing $A \oplus T$ for one of these patterns results in either the number 1 or 3. For a Q_2 pattern we obtain either 2 or 6, and for a Q_3 pattern either 5 or 7. Thus, T distinguishes between the different Q_i's. In particular, $c_1(A \oplus T) + c_3(A \oplus T)$ is a black and white image (as c_1 and c_3 are characteristic functions which set each pixel to zero unless it has value 1 or 3, respectively, in which case the new value is 1) where the number of black pixels corresponds to the number $n(Q_1)$. Continuing with this type of argument, it is not difficult to prove that the sum of the non-zero pixels in the image

$$C = c_1(A \oplus T) + 2c_2(A \oplus T) + c_3(A \oplus T) - c_5(A \oplus T) + 2c_6(A \oplus T) - c_7(A \oplus T)$$

represents the number $n(Q_1) + 2n(Q_2) - n(Q_3)$. An example is provided by Figure 2. Thus,

$$\chi = \frac{1}{4} C \cdot I = \frac{1}{4} \sum_{x \in X} c(x).$$

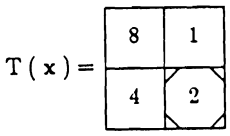
As a second observation we note that although the algebraic formula for expressing χ may seem a bit lengthy, $A \oplus T$ needs to be computed only once! The remaining operations for computing the image C are obviously local operations as addition is performed only between spatially corresponding pixels and the characteristic function determines the new state of a pixel in terms of its given state. The final dot product $C \cdot I$ can be computed by shifting the occurrences $c(x)$ across all the rows in parallel into the leftmost cells and then summing the result in the leftmost cells. The sums can then be shifted upward from these leftmost cells and summed in the uppermost left cell. Assuming that adding two numbers takes unit time, then the total time required for shifting and summing is proportional to the width plus height of the image, or of order $n[O(n)]$ for an nxn image. This method can be used to compute the dot product in $O(n)$ time. An alternative approach is the cellular pyramid architecture which allows for computation of the dot product in $O(\log n)$ time [11].

$$\underset{\text{(a)}}{\begin{array}{cccccc} 0&0&0&0&0&0\\ 0&1&1&1&1&0\\ 0&1&0&0&1&0\\ 0&1&0&0&1&0\\ 0&1&1&1&1&0\\ 0&0&0&0&0&0 \end{array}} \qquad \underset{\text{(b)}}{\begin{array}{cccccc} 3&4&4&4&1&0\\ 4&5&4&7&4&0\\ 4&4&0&4&4&0\\ 4&7&4&5&4&0\\ 1&4&4&4&3&0\\ 0&0&0&0&0&0 \end{array}} \qquad \underset{\text{(c)}}{\begin{array}{cccccc} 1&0&0&0&1&0\\ 0&-1&0&-1&0&0\\ 0&0&0&0&0&0\\ 0&-1&0&-1&0&0\\ 1&0&0&0&1&0\\ 0&0&0&0&0&0 \end{array}}$$

Figure 2. (a) The Boolean image A; (b) the image $A \oplus T$; (c) the image C.

Note that the sum of the entries of C equals zero which is the Euler characteristic of the "black" part of A.

It also follows from Gray's result that when using the template



the formula for computing the Euler characteristic reduces to the shorter formula:

$$\chi = [c_2(A \oplus T) - c_7(A \oplus T) + C_{10}(A \oplus T)] \cdot I.$$

The proof is similar to the one given previously. A short induction proof of the genus of a black and white image can be found in [13].

Computation of perimeter and area of black objects in Boolean images can also be achieved by bit quad pattern counting, and can be algebraically formulated in terms of the same template T as was used for the computation of χ [14]. Of course, Euler characteristic, boundary and interior can be rigorously defined in terms of the cellular topology used. Thinning on the other hand depends on the desired type of thinned image output. One thinning method--a variant of what is commonly known as the *medial axis transform*--is reminiscent of collapsing regular neighborhoods to their spines. The basic idea behind this thinning scheme is to find and label the centers of the largest disks that fit into the object to be thinned (shrumk) such that the boundary of the disk touches the boundary of the object in at least two points.

To be more precise, let A be a Boolean image and B(A) the black part of A. Let D(x) denote the "digital disk of radius 1" at location x, $D^2(x)$ the digital disk of radius 2, and, in general, $D^i(x)$ the digital disk of radius i (see Figure 3).

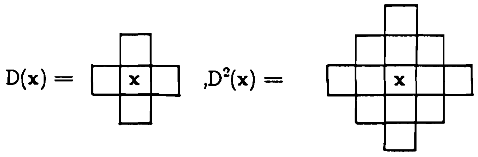
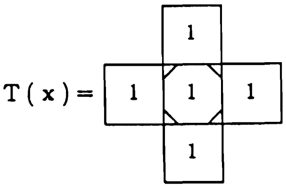


Figure 3. The shaded center cell corresponds to the location x.

The medial axis transform is then obtained by writing a program which computes the image $M = \{(x,m(x)): m(x) = k$ if $D^k(x) \subset B(A)$ and $D^j(y) \not\subset B(A)$ whenever $j > k$ and $y \in D(x)$, else $m(x) = 0\}$. The nonzero pixels of M are the weighted "medial axis" pixels of B(A). The medial axis pixel values correspond to the radii of the maximal disks and can be used to reconstruct *most* of B(A). The medial axis transform can be computed in parallel using neighborhood operations. In particular, if T denotes the template defined by



then the following short algorithm computes the medial axis transform in terms of the image algebra:

```
BEGIN
     $i = 0$
     $A_0 = A$
     DO UNTIL $B_i = 0$
          $A_{i+1} = A_i \owedge T$
          $B_i = A_{i+1} * c_0[(A_{i+1} \owedge T) \ovee T]$
          $i = i + 1$
     ENDDO
     $B = \sum_{k=1}^{i} kB_k$
```

Here A denotes the Boolean input image, c_0 the characteristic function which sets all zero values equal to one and all nonzero values to zero, and B represents the medial axis image. Figure 4 illustrates the medial axis for a very simple case.

$$\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & & & \text{(a)} & & &
\end{array}
\qquad
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 2 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 3 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 2 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & \text{(b)} & & &
\end{array}$$

Figure 4. The medial axis transform: (a) shows the black part of the Boolean input image A and (b) the essential part of the medial axis image B.

In higher dimensional cases ($X \subset Z^k$, $k > 2$) algorithm description and proof of algorithm correctness becomes a much more difficult and intricate task. Surface classification theorems, similar to Theorems 3.1 and 3.2 in the 2-dimensional case, have yet to be established. In many three dimensional imaging applications, the three-dimensional scene is represented by a three-dimensional array of pixels called *volume elements*, or *voxels* for short. An object B of voxels in the scene is specified by some property and it is often of interest to detect the surface of B for display purposes and analysis. An application area is computerized tomography which provides a representation of the human body by assigning density values to three-dimensional spatial locations. Organs can be distinguished from their immediate surroundings if the density value of voxels just inside the organ are different from those of adjacent voxels just outside the organ. The boundary between the organ and its surroundings can then be represented by a set of faces separating pairs of voxels. The faces are the intersecting faces of the cells representing the pairs of voxels.

Using classical three-dimensional topology, Dallas Webster and G. Herman provided a sequence of topological proofs which allow the detection of object surfaces in a computationally efficient way [15]. However, the proofs do not address the classification problem.

4. The Topology of Biological Vision Systems

Bionics is concerned with the study and design of machines that emulate biological systems. Thus, understanding the architecture and functions of the biological system to be modeled is of prime importance. Since the biological brain is the least understood system, it is not surprising that bionic vision is only in its primordial stage of development.

The importance of biological vision can be inferred from the fact that nature invented the eye at least three times. The cephalopod eye, the insect eye, and the vertebrate eye all have totally different and independent evolutionary histories. Nevertheless, these histories have converged to essentially the same result. The neural networks in each of these eyes are surprisingly similar. The basic components of these living image processing systems can be represented by a block diagram containing four main elements as illustrated in Figure 5. The main elements are: a sensor for image acquisition, a preprocessing element for image enhancement, filtering and image transformation, a processing unit for the analysis, recognition and interpretation of the sensed image, and a memory which may be dynamic and/or static (genetically imprinted) for referencing and possibly storing image information. This does not mean that sensed images are processed and perceived in a similar manner. The processing and perception of, and responses to, sensed images vary greatly between different species. For instance, octopi and humans do *not* see the same things. Octopi cannot distinguish between mirror images.

$$\begin{tikzcd}
\text{Image} \arrow[r] & \boxed{\text{Sensor}} \arrow[r] & \boxed{\text{Preprocessor}} \arrow[r] & \boxed{\text{Processor}} \arrow[r, leftrightarrow] \arrow[d] & \boxed{\text{Memory}} \\
& & & \text{Response} &
\end{tikzcd}$$

Figure 5. The four basic components of an image processing system.

As an image processing system, the human eye and visual cortex are unsurpassed. Humans are capable of recognizing a myriad of objects in his visible environment as well as artificial abstractions of these objects. The human visual system uses over 120,000,000 sensing elements in each eye. Preprocessing of the raw sensed image takes place in the second and third layer neurons of the retina and the neurons of the geniculate body. The neurons of the geniculate body are connected to the retinal cells of the optic nerve bundle which consists of approximately 800,000 nerve fibers. The geniculate body transmits the transformed images to the 6,000,000,000 neurons of the visual cortex that are directly involved with image perception. Although the mechanisms have only been hypothesized, neurons may have the capacity of storing billions of bits of information. Neurons have both analog and digital properties. Input and processing consists of graded potentials, while output consists of fixed voltage, all-or-none pulse trains.

The neurons of the visual cortex are organized into columns of up to a hundred thousand neurons per column. Processing in columns proceeds in parallel. For example, a portion of an image can be processed in parallel by columns which extract lines, edges and other features. Columns are arranged with inputs from other columns at many layers. This allows multiple overlays of two-dimensional patterns in the horizontal plane while maintaining the topology of connections with other areas [16,17].

Even the largest computers and novel *connection* machines, using VLSI technology in order to integrate 100,000 transistor devices per chip, are minute in comparison to this enormous neural network. It should therefore not be surprising that current computer based image processing systems are still a long way from being capable of recognizing the wide variety of objects that any ordinary human being can recognize. However, the human visual system serves as an excellent example that general purpose image processing systems can be built. It should be clear that this emulation will be more of style than of detail. VLSI devices will remain in a different class from that of biological vision systems. We should not expect--or even strive for--the exact machine duplication of a biological system. What can be expected is the design of machines that emulate and surpass a particular biological vision system in the performance of many tasks while being upstaged by that biological system in the performance of various other tasks. It is very much like the "airplane versus bird" analogy. Airplanes are a direct result of man's attempt to mimic birds. They outperform birds in terms of speed and carrying capacity and lag far behind a bird's maneuverability and flight safety.

As a first attempt in modeling the vertebrate visual system, we shall only consider the sensor and preprocessing components. In the human visual system, the sensor is the eye which is shaped like a sphere. The retinal layer of the eye consists of three sets of neurons arranged in layers. The innermost layer contains the rod and cones which are the receptors of sight. The part of the outside world seen by *one* eye at any moment is called the *visual field* of that eye. Due to the restriction of the visual field caused by the nose, brow, and cheek, the stereographic projection of the retina into a plane does not result in a circular visual field.

When light strikes the retinal receptors, impulses are set up and transmitted to the nerve cells of the other layers, called the first neurons and second neurons. The second neurons are also called the ganglions. The series of cells, receptors, first neurons and second neurons, and their synapses do not form a simple bucket brigade of impulses. A receptor may send impulses along its axon and dendrites to more than one cell of the first neuron and several receptors may synapse with the same cell. The axons of the ganglion cells form the *optic nerve*. There is general agreement among physiologists that the principal function of the first and second nerve cells of the retina is probably to compress the information contained in the activity of a very large number of receptors into a much smaller number of channels, the fibers of the optic nerve. In doing this, a great deal of information is being discarded. This loss of information is necessitated by the compression into fewer channels. Some discarding can be advantageous if biologically important information is to be efficiently sifted from unimportant information. There can be little doubt that it is easier for a brain to deal with little information than with too much. For our purposes it is, therefore, not unreasonable to assume that the first and second neurons are part of an imaging preprocessing system which compresses and filters information [18,17].

The fibers of the optic nerve that supply visual information to the cerebral cortex do not pass directly to it, but synapse with the cells in the lateral geniculate body. Several theories as to the function of the geniculate body have been proposed. The most widely accepted hypothesis is that the neurons of the geniculate body compress and translate received information into a code that can be accepted by the neurons of the cerebral cortex. Thus, the geniculate body may be viewed as a preprocessing system which sets the stage for the image understanding system of the cerebral cortex.

The retina is mapped in a regular manner to the lateral geniculate body. Cells that are spatially close in the retina fire neurons in the geniculate body that are near to each other. The mapping of the retina on the cat's and rhesus monkey's geniculate body have been thoroughly examined by electrophysiological methods. In particular, the maps preserve spatial continuity both from the retina to the geniculate body and from the geniculate body to the retina. The area of a receptor moasic in the retina feeding into a single cell of the geniculate is called the *receptive field* of that cell. The receptive field of a geniculate cell consists of a small disk-like mosaic of retinal cells. This disk is very small near the *fovea centralis*, the center of direct vision, and increases in size toward the outer edge of the retina which is involved with peripheral vision. It is also well known that the receptive disk is the union of two functionally distinct regions, consisting of a smaller central disk and a surrounding annulus.

One of these regions is called an "on" region and the other an "off" region. Depending on which region a spot of light falls, either of two responses could be produced. The firing rate of cells in the "on" region is increased under the stimulus of light, while in the "off" region the stimulus of light deccreases the cell's firing rate. The two effects tend to neutralize each other. When both the center and the surround are stimulated together, the antagonistic input sum-- in a still unspecified, but most likely algebraic way-- to produce the next result [19,20,21].

One important consequence of this "on"-"off" organization of the receptor disks can be demonstrated by using large stimulus figures. For example, consider a bright white figure on a black background. If the figure covers the whole receptive disk, a weak response is elicited because both the "on" and "off" regions are stimulated. But if the figure is positioned so that the black contrast border falls just to one side of the smaller central disk region, and if this region is an "on" region, then the surrounding annulus is not stimulated as strongly and the response is increased. This phenomenon corresponds to edge filtering in image processing by computer.

The cellular topology and algebraic structure discussed in the previous sections provide a rather natural interpretation of the retino-geniculate structure as a bionic system. Suppose X denotes the planar projection of the right visual field into the plane. Since X is discrete, we assume further that $X \subset Z^2$. Each point of X corresponds to a retinal receptor. Let Y denote neurons of the geniculate body having synaptic connections with the cells of X. In comparison to X, which is "flat," Y is 3-dimensional. For this reason we think of Y as a subset of Z^3.

There are several choices for defining topologies on X and Y. Since the von Neumann topology is particularly simple and actually provides for the close mimicking of the biological visual pathway, we endow both X and Y with the von Neumann topology. Now let $\mathcal{J}(y) \subset X$ denote the receptive field of $y \in Y$. Recall that $\mathcal{J}(y)$ is a small disk-shaped mosaic of sensors. Thus, it is reasonable to let $\mathcal{J}(y) = D^i(x_y)$, where $D^i(x_y)$ is a digital disk of radius i and center x_y. If $i = 0$, then $\mathcal{J}(y) = x_y$. We also assume that i is even if the sum of coordinates of y is odd, and i is odd if the sum of coordinates is even.

Defining the set $t_y = \{(x,t_y(x)): t_y(x) =$ the contribution of x on the firing or on inhibiting the firing of $y\}$ defines a template $T = (\mathcal{J},t)$ from Y to X. We assume of course that $t_y(x)$ is a determinable numeric quantity for each pair x and y. Note also that $t_y(x) = 0$ whenever $x \notin \mathcal{J}(y)$.

Now consider the stimulus response of the retina to a bright white figure on a black background. Let $A = \{(x,a(x)): a(x) = 1$ if x is fired by the figure, else $a(x) = 0\}$. The geniculate response can then be interpreted as

$$A \oplus T = \{(y,c(y)): c(y) = \textstyle\sum_{x \in \mathcal{J}(y)} a(x)t_y(x), y \in Y\}$$

For simplicity we suppose that $A \oplus T$ is Boolean, that is $c(y) = 1$ if the sum $\sum a(x)t_y(x)$ exceeds the firing threshold, else $c(y) = 0$. This supposition can easily be obtained by the operation $c_{\geq k}(A \oplus T)$, where k denotes the firing threshold.

Let $B(A \oplus T) = \{y \in Y: c(y) = 1\}$ and define a function $f: B(A \oplus T) \to \mathcal{J}(B(A \oplus T))$ by $f(y) = x_y$, where x_y denotes the center of $\mathcal{J}(y)$. It then follows from the definition of $\mathcal{J}$ that f is continuous and, in fact, open. Thus, at least near the fovea, f is an embedding. Hence, what is "perceived" by the geniculate body corresponds to a "fattened-up" version of $B(A) = \{x \in X: a(x) = 1\}$, where the "fattening" corresponds to covering B(A) by the receptive disks. In particular, if B(A) corresponds to such figures as a digital line, disk, or circle of sufficiently large diameter, then $f(B(A \oplus T))$ is a deformation retract of $\mathcal{J}((B(A \oplus T))$ and

(I) $\pi_1(B(A \oplus T)) = \pi_1(f(B(A \oplus T))) = \pi_1(\mathcal{J}(B(A \oplus T)))$.

As a line and disk are (digitally) convex so are their covers by small digital disks, in which case we have that $\pi_1(\mathcal{J}(B(A \oplus T)) = \pi_1(B(A))$. The same situation holds for circles of sufficiently large diameters. Thus, under the right conditions of visual resolution, equation (I) implies

(II) $\pi_1(B(A \oplus T)) = \pi_1(B(A))$.

Hence, the object sensed by the retina is homotopically equivalent to the object perceived by the geniculate.

A somewhat similar but weaker result was obtained by E. C. Zeeman in his classical paper on the topology of the brain [22]. He showed that the right visual field X and the right *visual lobe* Y have isomorphic homology theories.

In particular, ignoring the geniculate body, and using his tolerance topology on X and Y, this result implies the Czech homology groups for A and $\Lambda \oplus T$ are isomorphic.

The insistence on circles with sufficiently large diameter has to do with the eye's *visual resolution* or *visual acuity*. Visual acuity refers to the eye's ability to determine the precise shape or detail of an object, or recognize the separateness of two small objects placed close together. This ability is exhibited in the highest degree near the fovea centralis. Pairs of points that are indistinguishable by the eye are said to be within visual acuity tolerance. In looking at a straight line, the eye can detect a lateral break that forms an image only 1×10^{-5} cm wide. This corresponds to approximately a 30-th of the diameter of a retinal receptor. This precision, 30 times finer than the size of a receptor cell, seems to be due to the tiny scanning motions, or *saccades*, that are necessary for keeping a static pattern in view even over a short period of time [23].

It would be interesting and extremely worthwhile to attempt to model a mosaic of receptors, much like the region of the fovea, complete with scanning motions, that is capable of similar accuracy. This model would probably involve some group theory. Fuzzy cellular topologies instead of rigid ones might provide an architectural description that is more in tune with the actual biological system. Current multi-aperture mosaics, based on the retina of the insect eye, have not incorporated saccade-like scanning motions and have extremely poor resolving power [24].

A great deal of work remains to be done to establish a solid theoretical and practical foundation of bionic vision systems. The topic not discussed in this section, namely image interpretation and understanding, will without doubt remain a major obstacle to this goal. To this day, only a few bionic vision systems have been built, generally in computer vision experiments. However, since biological vision systems are working examples of massively parallel, densely interconnected computational networks, crossfertilization of knowledge of biological and computer based systems will no doubt remain as active and important in the future as it is in the present.

5. Simplicial Codes

In order to analyze, synthesize, and manipulate geometric configurations by means of a digital computer, the need arises for precise methods of describing these configurations. For example, if one wishes to transmit the surface contour of an airplane over a communication link, the contour must first be described and encoded in a fashion that permits efficient transmission. The decoding at the receiving end should permit a faithful pictorial reconstruction of the airplane's surface.

One of the simplest and most useful methods which permits the encoding of arbitrary digital planar curves is known as the *octagonal chain code* [25]. Suppose we have a black digital curve on a white background with $X \subset Z^2$. If a point on the curve is known, then the "next" point can assume only one of eight possible adjacent positions as shown in Figure 6. If one assigns the integers zero through seven to these eight positions, starting with the one which is horizontally to the right and progressing in a counter-clockwise direction, the code shown in Figure 7(b) is obtained from the digital curve in Figure 7(a) if the starting point is the pixel closest to the upper left hand corner of the image.

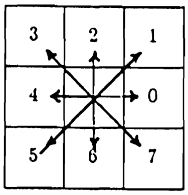
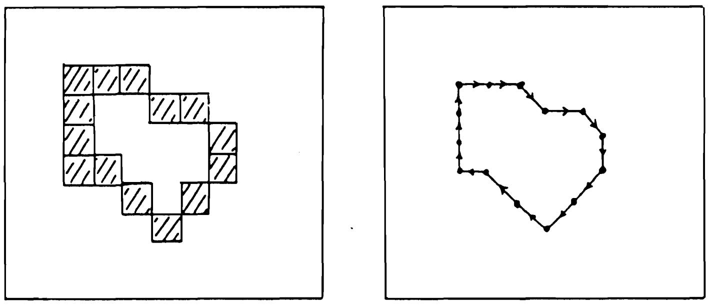


Figure 6



(a) (b)

The chain code 00707655334222

Figure 7

The reconstruction of the image from its contour code is straight forward, given an understanding of the mapping strategy. Given the starting point, we simply fill in the cells sequentially according to the direction given by each element of the code, starting with the left most integer of the code. Given the code without a starting point, the original curve can still be faithfully reconstructed modulo a shift in location. Using clever programming techniques, Boolean images containing many digital simple and non-simple (self-intersecting) curves can be encoded using the basic eight direction scheme. One important observation about the octagonal chain code is the fact that only three bits are required to specify one point on the curve; i.e. in binary form 0 equals 000, 1 equals 001, 7 equals 111, etc. The required memory capacity for a curve encoded in this manner is then only 15% of that required for a curve which has all its points independently specified in a 1024×1024 point array.

However what makes the octagonal chain code extremely interesting is not necessarily its simplicity and data reduction capability, but rather its manipulative properties. That is, if the code for the boundary of an object is known, then it is possible to compute such measures as area, perimeter, center of gravity, moments, maximum height, maximum width, homotopy, and so on, directly from the code [25]. The number of self-intersection points and whether or not the curve is closed or not can also be directly determined from the code. Another interesting fact is that addition of 2 (mod 8) to each integer in the code causes the curve represented by the code to be rotated counter-clockwise by 90°. Doubling the number of odd digits in the code representing the boundary of a figure and then adding

1 (mod 8) to each integer in the code, will rotate the figure and double its size. The doubling is necessary in order to preserve connectivity properties.

From a topologist's point of view it is interesting to note that the chain code is essentially an oriented simplicial 1-complex, each letter representing a 1-simplex with a given orientation (see Figure 7(b)). However, as a 1-complex it can not be employed for coding digitized 2-dimensional regions. In particular, the 1-complex fails to relate information that is independent upon the contour of a 2-dimensional region. Information as to coloration and different depths could never be retrieved from any type of contour encoding.

One method of encoding digital surfaces in two or three dimensional space consists of generalizing the concept of the octagonal chain code to oriented 2-dimensional simplicial complexes. In order to simplify our discussion, we present the code for planar digital surfaces and refer the interested reader to [26] for an in-depth discussion of encoding surfaces in 3-dimensional space.

When subdividing a digital surface into a simplicial complex, we will use only the four basic triangles shown in Figure 8. We agree that each triangle has a counterclockwise orientation. Thus, using the standard octagonal code, the orientation of the 2-simplex labeled 0 is given by 614; 1 has orientation 036; 2 has orientation 250; and 3 has orientation 472. Each triangle has a vertical, a horizontal, and a diagonal (hypotenuse) side. We denote the vertical side by V, and horizontal side by H, and assume that each has unit length.

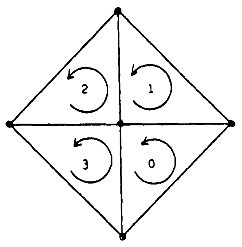


Figure 8

If a and b are two triangles, then the symbol aVb means that a and b are attached along their vertical sides. Similarly, aHb and ab will mean that a and b are attached along their horizontal and along their diagonal sides, respectively. The 10 possible combinations are shown in Figure 9. Several observations are now in order. First, the operation of attaching is commutative, that is 0H2 = 2H0. Second, the attachment results in a coherently oriented simplicial complex. Thus, when substituting the octagonal codes for the boundaries of the triangles, the interior edge cancels and we obtain the boundary of the simplicial complex defined by the two triangles. For example, 0H2 = (614)(025) = 61/4/025 = 6125. Note that we write 0 and 2 so that their joining edges, H = "4" and H = "0," respectively, appear in juxtaposition. This observation can be used to obtain the octagonal chain code of the boundary from the two dimensional simplicial code of the surface. Finally, observe that some combinations such as 12 and 1H2 are impossible or do not yield simplicial complexes.

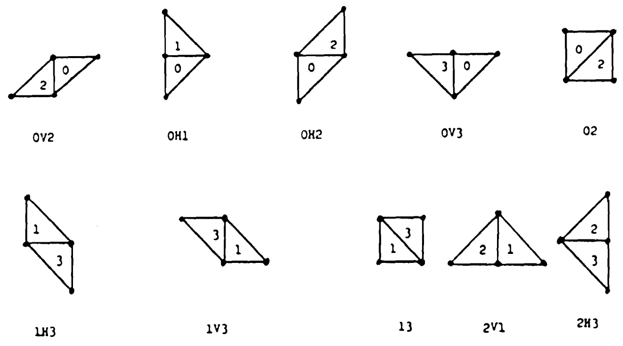


Figure 9

We are now in the position to define the grammar of our code. The triangles 0, 1, and 2, and the edges H and V are called *simplicial letters*. Any finite string of simplicial letters written in juxtaposition, with triangles and attaching rules alternating, is called a *simplicial word*. A simplicial word can be *realized* as a simplicial complex if the following conditions are satisfied:

(1) The first and the last letter in the word are triangles.

(2) Attaching the triangles via the attaching rules, in the order of occurrence when the word is read from left to right, yields a simplicial complex.

(3) When attaching the triangles, no triangle is superimposed on a previously attached triangle.

A *simplicial sentence* is a simplicial word which can be realized as a simplicial complex. Thus 1H02V1H3 is a simplicial sentence, while 1H02V1H3H2 and 1H02V1H3H13V1 are not. In the last case we note that the 6th and 10th letters are superimposed. Obviously, the 10 combinations given in Figure 9 are all simplicial sentences as well as the simplicial letters 0, 1, 2, and 3.

In order to realize a large variety of simplicial complexes, we need to introduce the notion of attaching a word to a sentence. Let $S_1, S_2, \ldots, S_k$ and $T_1, T_2, \ldots, T_n$ be simplicial letters, $S = S_1 S_2 \ldots S_k$ a simplicial sentence, and $T = T_1 T_2 \cdots T_n$ a simplicial word. We say that $\mathcal{T}$ *can be attached to* S *at* S_i if $S_1 S_2 \ldots S_i T_1 T_2 \cdots T_n$ can be realized as a simplicial complex so that the triangles determined by T do not intersect the interiors of the triangles of S determined by $S_{i+1} \ldots S_k$. If T can be attached to S at S_i, then we write

$$S^i \star T \equiv S_1 S_2 \cdots S_i (S_{i+1} \cdots S_k) T_1 T_2 \cdots T_n$$

and call $S^i \star T$ the *simplicial sentence obtained by attaching* T *to* S *at* S^i. Thus, $S^i \star T$ determines a simplicial complex which can be realized by first realizing the complex determined by S and then attaching the triangles of T to the triangles S by using $S_i T_1 T_2 \cdots T_n$ as our sentence of instructions. Observe that if i = k, then $S^i \star T \equiv S_1 S_2 \cdots S_k T_1 T_2 \cdots T_n$ is a word without parenthetic letters.

We include the parenthetic symbols "(" and ")" in our set of simplicial letters, and note that our simplicial alphabet contains $2^3 = 8$ letters.

Sentences which include parentheses are also called *complex sentences*, while those which do not are called *simple sentences*. We shall enlarge our class of simplicial words by including sequences of letters which are obtained from complex sentences by the deletion of either the first or the last letter, or both. The concept of attaching a word to a complex sentence is analogous to the concept of attaching a word to a simple sentence. In particular, if U is a complex sentence and W a word, then $U^i * W$ means to attach the triangles of W to the triangles of U using U_iW as the sequence of instructions, where U_i denotes the ith letter in the sentence U. Again if $U^i * W$ can be realized as a simplicial complex, then $U^i * W$ is referred to as a simplicial sentence. For an algorithmic implementation of this code and further examples, we refer the interested reader to [26].

Since the simplicial 2-complex code represents the 2-dimensional generalization of the octagonal chain code, it inherits many of the simple and powerful manipulative powers of its one dimensional predecessor. For example, addition (mod 4) to a triangle rotates the triangle through an angle of 90° counterclockwise. Forty-five degree rotations are again obtained via expansion of the code. Area determination is particularly easy. It is simply 1/2 times the number of triangles appearing in the code.

Coding non-planar surfaces such as digital spheres or tori in 3-space is somewhat more involved. However, the basic idea is the same as in the lower dimensional cases. It consists of attaching oriented 2-simplexes to grid points in discrete 3-space in a coherent fashion. Of course, since a point $x \in X \subset Z^3$ has 26 grid neighbors, the number of letters of the coding alphabet increases accordingly. In particular, $2^4 = 16$ letters (symbols) are needed to encode any digital surface in 3-space if the generalization of the octagonal chain code is used [27]. It has been conjectured (but not proven) that 2^{n+1} letters suffice to encode an (n-1)-dimensional surface in n-space if the code represents the natural generalization of the above described simplicial scheme.

Although it requires only 3 bits to encode a digital curve or surface point in Z^2 and four bits in Z^3, the simplicial codes are not as efficient as they appear at first glance. Simplicial sentences describing digital figures can be unnecessarily long and difficult to read since a large number of simplexes are necessary to describe even the simplest configurations such as a long line or a large rectangle. For example, a horizontal line consisting of n+1 pixels could be more easily represented by a symbol such as 0_n instead of a sequence of n zeros. Using oriented cell complexes consisting of cells having regular shapes but varying sizes may be one possible approach to encode digital surfaces. A large rectangle should be representable by a single cell. The theoretic and algorithmic definition of such a code could be a major contribution to image processing and would certainly be a worthwhile undertaking.

There are many other topics that could have been included in this paper. There exists a considerable amount of literature on convexity; on shrinking; on homotopy and dimension [4]. The study of geometric properties, known as *mathematical morphology* and based on Minkowski's geometric measure theory has rapidly grown during the past decade

[28]. Hausdorff dimension and fractal theory for modeling natural scenes are starting to play an important role [29]. Finding fractal invariants is considered an important and difficult problem. If these topological problems help generate sufficient interest among topologists, then the intent of this paper will have been fulfilled.

6. Acknowledgements

This work has been partially supported by the Air Force Armament Division and DARPA under Contract F08635-84-C-0295. The author is especially thankful to Dr. Sam Lambert and Mr. Neal Urquhart of the Air Force Armament Laboratory and to Dr. Jasper Lupo of DARPA for their continued support of this research.

References

1. J. von Neumann, *Theory of self-reproducing automata*, University of Illinois Press, Urbana, IL (1966).
2. E. F. Moore, *Machine models of self-reproduction*, AMS Proc. of Symposia in Applied Math. 14 (1961).
3. L. A. Ankeney and G. X. Ritter, *Cellular topology applications in image processing*, Internatl. Journ. of Comp. and Inf. Science 12(6) (1983), 433-456.
4. A Rosenfeld, *Digital topology*, American Math Monthly 86 (1979).
5. ________, *Arcs and curves in digital pictures*, J. Assoc. Comput. Machines 20 (1976).
6. S. H. Unger, *A computer oriented toward spatial problems*, Proc. IRE 46 (1958).
7. A. W. Burks (ed.), *Essays on cellular automata*, University of Illinois Press, Urbana, IL (1970).
8. K. E. Batcher, *Design of a massively parallel processor*, IEEE Trans. Computers 29(9) (1980).

9. E. Cloud and W. Holsztynski, *Higher efficiency for parallel processors*, Proc. IEEE Southcon 84 (March, 1984), 416-422.
10. M. Duff, CLIP4 *a large scale integrated circuit array parallel processor*, in 3rd Internatl. Joint Conf. on Pattern Recognition (1976).
11. G. X. Ritter and P. D. Gader, *Image algebra implementation on cellular array computers*, IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, Miami Beach, FL (1985), 430-438.
12. S. B. Gray, *Local properties of binary images in two dimensions*, IEEE Trans. Computers 20 (1971).
13. A. Rosenfeld and A. C. Kak, *Digital image processing*, Academic Press, New York (1976).
14. W. K. Pratt, *Digital image processing*, John Wiley, New York (1978).
15. G. T. Herman and D. Webster, *A topological proof of a surface tracking algorithm*, Computer Vision, Graphics, and Image Processing 23 (1983).
16. B. G. Cragg, *The topography of the afferent projections in the circumstriate visual cortex of the monkey studied by the Nanta method*, Vision Res. 9 (1969).
17. G. Shepard, *The synaptic organization of the brain*, Oxford Press, New York (1963).
18. E. Gardner, *Fundamentals of neurology*, Saunders Co., Philadelphia (1963).
19. D. H. Hubel and T. N. Wiesel, *Receptive fields of optic nerve fibers in the spider monkey*, Journ. Physiol. 154 (1960).
20. R. L. DeValois and P. L. Pease, *Contours and contrast: responses of monkey lateral geniculate nucleus cells to luminance of color figures*, Science 171 (1971).
21. G. H. Jacobs and R. L. Yolton, *Distribution of excitation and inhibition in receptive fields of lateral geniculate neurons*, Nature 217 (1968).

22. E. C. Zeeman, *The topology of the brain and visual perception*, in Topology of 3-manifolds, M. K. Fort, Jr. (ed.), Prentice-Hall, Englewood Cliffs, NJ (1962).

23. J. R. Platt, *How we see straight lines*, Scientific American 202 (1960).

24. R. T. Schneider, J. D. Cox and J. Ahmad, *Pattern recognition for multiaperture optics systems using low pixel numbers*, Proc. Soc. of Photo-Optical Instrumentation Engineers (1984), 504.

25. H. Freeman, *On the encoding of arbitrary geometric configureations*, IRE Trans. Electron. Comput. 10 (1961).

26. S. M. Boyles and G. X. Ritter, *The encoding of arbitrary two-dimensional geometric configurations*, Internatl. Journ. of Comp. and Inf. Science 10(1) (1981).

27. G. X. Ritter and J. T. Tou, *The encoding of arbitrary surfaces in 3-dimensional space*, Pattern Recognition 17(6) (1984).

28. J. Serra, *Image analysis and mathematical morphology*, Academic Press, London (1982).

29. A. P. Pentland, *Fractal-based description of natural scenes*, IEEE Trans. on Pattern Analysis and Mach. Int. 6(6) (1984).

University of Florida
Gainesville, Florida 32611